

Nro. ord.	Apellido y nombre	L.U.	#hojas

ORGANIZACIÓN DEL COMPUTADOR I - Parcial

Primer Cuatrimestre 2019 - Turno Tarde

Ej.1	Ej.2	Ej.3	Ej.4	Nota

Corrector:

Aclaraciones

- Anote apellido, nombre, LU y numere *todas* las hojas entregadas
- Cada ejercicio se califica con Bien, Regular o Mal. La división de los ejercicios en incisos es meramente orientativa. Los ejercicios se califican globalmente.
- El parcial **NO** es a libro abierto, pero puede tener los apuntes provistos por la cátedra y una hoja A4 con apuntes propios
- **Importante:** Justifique sus respuestas. Las soluciones a ejercicios de la práctica que se utilicen deben ser incluidas en el examen.
- El parcial se aprueba con un Bien y a lo sumo un Regular en los ejercicios 1, 2. En los ejercicios 3 y 4 al menos uno Bien.

Ejercicio 1 La ATR es una máquina con arquitectura Von Neumann que opera con palabras e instrucciones de tamaño fijo y utiliza aritmética en complemento a 2. Tiene 32 registros de propósito general y una memoria direccionable a *byte* con direcciones de 12 *bits* y con palabras de 16 *bits*.

El set de instrucciones es el siguiente:

Instrucción	Formato	opCode	Efecto
MOV regX, regY	RR	000	regX \leftarrow regY
ADD regX, regY	RR	001	regX \leftarrow regX + regY
NEG regX	R	010	regX \leftarrow !regX (not bit a bit)
JZ inm	RJ	011	si Z = 1, PC \leftarrow PC + ext(inm)
JN inm	RJ	100	si N = 1, PC \leftarrow PC + ext(inm)
LD reg, inm	RM	101	reg \leftarrow [inm]
ST reg, inm	RM	110	[inm] \leftarrow reg
JMP inm	M	111	PC \leftarrow inm

En todos los casos, la referencia a PC corresponde al valor del mismo al realizar la ejecución de la instrucción. ext() se refiere a extender el signo hasta el tamaño adecuado. Las únicas instrucciones que afectan los *flags* son ADD y NEG. LD y ST sólo utilizan R0 o R1

Las instrucciones tienen 4 tipos de formatos posibles:

<i>bits</i>	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	3 bits			3 bits			5 bits			5 bits						
RR	opCode			0			reg			reg						
	3 bits			8 bits			5 bits									
R	opCode			0			reg									
	3 bits			5 bits			8 bits									
RJ	opCode			0			inm									
	3 bits			1 bit			12 bits									
RM	opCode			reg			inm									
	3 bits			1 bit			12 bits									
M	opCode			0			inm									

a. Definir (justificando su respuesta):

- El tamaño del PC
- El tamaño máximo de la memoria
- Cuántas instrucciones sin operandos podría agregar manteniendo las instrucciones actuales (Tenga cuidado que las instrucciones actuales sigan siendo posible de decodificar).
- Considerando que una instrucción inválida es una tira de *bits* que no codifica una instrucción que la máquina puede ejecutar, determinar de las siguientes codificaciones cuáles lo son: 0x29, 0xEACA, 0xCACA, 0x24AA, 0x8038.

- b. Se quiere ensamblar y cargar desde la posición de memoria 0x000 el siguiente código en una ATR:

```
inicio: LD R0, 0x00A
        ADD R0, R0
        LD R1, 0x0016
        MOV R2, R0
        NEG R2
        ADD R2, R1
seguir:  ADD R0, R2
        JZ fin
        JMP seguir
fin:     ST R0, 0x00C
        DW 0x114A
        DW 0x0001
```

- I. Definir a qué posición de memoria corresponde cada etiqueta.
 - II. Indicar el contenido de la memoria luego de ensamblar y cargar el código anterior.
- c. Para cada parte de la planilla de seguimiento de la máquina ORGA1, justificar si es necesaria o no dicha celda. Indicar si es necesario agregar nuevas celdas o no para poder realizar el seguimiento.
- d. Modificar la planilla de seguimiento de la ORGA1 y realizar el seguimiento de la ATR. Tomar en cuenta que los que los registros de propósito general empiezan en cero y la memoria se carga inicialmente totalmente en cero.
- e. Mostrar el contenido de la memoria luego de la ejecución del programa.

Ejercicio 2

- a. Realizar el circuito del componente NEG, que toma un valor de 16 bits y devuelve el mismo valor negado bit a bit.
- b. Realizar el diagrama del *datapath* de una microarquitectura para la ATR que soporte la ejecución de las instrucciones descritas. Recuerde indicar el tamaño de cada registro, de los buses internos y externos, las señales de cada componente y justificar la utilización de cada componente escogido y cada decisión tomada.

Para realizar el *datapath* puede utilizar los siguientes componentes:

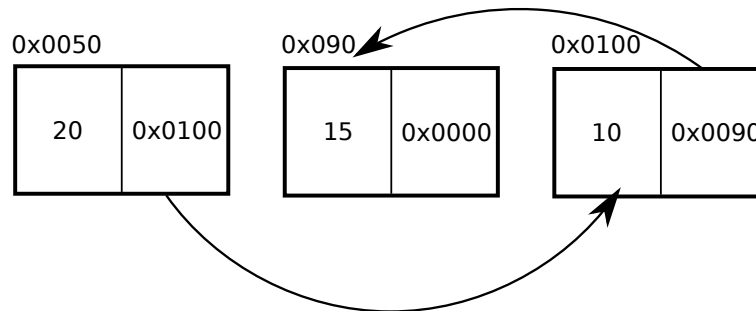
- una única ALU que realiza las operaciones suma y resta **con flags**.
- extensores de signo y componentes para completar con ceros. (Deberá indicar claramente el tamaño de los registros de entrada y salida)
- un único controlador de memoria.
- el componente del apartado anterior (Si no lo realizó puede suponerlo hecho)

Al incluirlos detallar cuidadosamente el tamaño de los registros y los nombres de las señales. Cualquier otro componente a utilizar deberá ser implementado e incluido en el examen.

- c. Escribir las microinstrucciones que debe ejecutar la máquina para realizar el *fetch* de una instrucción (no incluir etapas posteriores del ciclo de instrucción).
- d. Escribir el microprograma que realiza la parte de ejecución del ciclo de instrucción de las siguientes instrucciones:

- I. LD R0, 0x014
- II. JZ 0x0C
- III. JMP 0x000

Ejercicio 3 Se cuenta con una máquina ORGA1i. Se cuenta con la siguiente estructura (se indica cada celda en qué posición de memoria se encuentra):



La misma es una lista enlazada en la que cada celda contiene un número de 16 bits y luego otro número de 16 bits referenciando al siguiente elemento en la lista. El último elemento de la misma será el referencia a la posición 0x0000.

Realizar una función que tome en R0 la posición de memoria del primer elemento de una estructura. La misma deberá recorrerla sumando en cada paso el índice recorrido + el número en la posición. Devolver el resultado en R1.

En el ejemplo dado, el número final deberá ser: $(20 + 0) + (10 + 1) + (15 + 2) = 48$

Ejercicio 4 Se tiene el famoso y archiconocido juego *Western Bar*¹ multijugador. En dicho juego se tienen dos contrincantes. El objetivo es que un jugador mate al otro de un disparo de su Colt dragoon². El duelo comenzará al comenzar a sonar una campana (que podemos asumir que terminará antes que termine el juego) avisando que se puede comenzar a disparar. Mientras no se esté presionando el botón de disparo, los jugadores estarán detrás de una mesa, resguardados. Las balas van volando hasta el otro jugador y, si en ese momento el otro jugador está con el botón presionado, lo matará perdiendo así la partida. Esto se repite infinitamente hasta que nos cansemos y compremos la nueva PS5. Si alguien dispara en el tiempo entre el fin de una partida y el inicio de otra (la campana que comienza a sonar), ese disparo no será tenido en cuenta. El sistema deberá mantener en R5 la cantidad de veces que ganó el jugador 1 y en R6 la cantidad de veces que lo hizo el jugador 2.

Se cuenta con una ORGA1i y una rutina en el label **ayQueMePilla** que toma en R0 el tiempo en el que el jugador 1 disparó, en R1 el tiempo en el que el jugador 2 disparó, en R2 si el jugador 1 está con el botón apretado y en R3 si el jugador 2 está con el botón apretado. Dicha rutina tomará todo esto en cuenta devolverá en R4 el número del jugador que ganó el juego (1 o 2) o 0 si todavía ninguno lo hizo. Además se cuenta con algunos dispositivos de E/S detallados a continuación:

CAMPANA: informa si suena o no la campana. Indicará con el valor 0x0001 cuando comienza a sonar la misma y se pondrá en 0x0000 cuando deje de hacerlo.

TIEMPO: informa el tiempo en segundos. Arrancará en cero al prender la máquina y externamente se va incrementando.

BOTON1: informa el estado del botón 1. Si el mismo está apretado contendrá el valor un 0x0001 si no 0x0000.

BOTON2: informa el estado del botón 1. Si el mismo está apretado contendrá el valor un 0x0001 si no 0x0000.

Cada vez que BOTON1 o BOTON2 cambian su estado, INTR se pondrá en 1 y mantiene su valor hasta recibir un pulso en la entrada de un *bit* llamada INTA.

Se puede asumir que nunca llega a terminarse el timer del tiempo.

- Mapear los registro de E/S e indicar para cada uno si es de lectura, escritura o lectura/escritura.
- Realizar el esquema de conexión del sistema, incluyendo todos los dispositivos involucrados y la manera en la cual están conectados.
- Realizar el pseudo-código de la rutina principal y de la RAI que se debe cargar en la máquina ORGA1i para lograr el funcionamiento detallado.
- Realizar el código, en lenguaje ensamblador, de la rutina principal y de la RAI.

¹<http://www.handheldmuseum.com/Casio/WesternBar.htm>

²Si, somos muy pacifistas en la materia

